Casper Beginner User Manual

Staff of HPC@Polito - October 16, 2015

This guide is written for all users of the cluster
to answer all frequently asked questions.
If you need more information
don't hesitate to contact us at
hpc.dauin@polito.it

Gentlemen Agreements

- By using our systems for research purpose, you automatically authorize hpc@polito's staff to publish your personal data (name, surname, research group) and data associated with your research on our web site (hpc.polito.it) and in all the other papers published by hpc@polito (annual reports, presentations, etc.).
- By using our systems for research purpose, you agree to quote the hpc@polito's center in all your scientific articles on paper, conference or books. We kindly suggest this kind of acknowledgement:

  "Computational resources provided by hpc@polito, which is a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino (http://www.hpc.polito.it)".

  Or a shorter version as
  "Computational resources provided by hpc@polito (http://www.hpc.polito.it)".

Rules for Cluster Uses

- It is NOT permitted to run commands like simulations and computations directly from the command line, due to existing risk to make unusable *login-node* or other shared resources (by running such commands your account will be blocked). You MUST always pass through scheduler's computational queues (also for compilation purpose).
- Any uses of shared resources except didactical or research purposes are NOT permitted.
- Every user is responsible for activity done by his account, it is not recommended to share your account credentials with others, it is just allowed by previously informing HPC staff;

  Generally, it's NOT permitted to share sensible data, especially username and password, with others. The user agree to hold them safely.
- As a general rule, it is NOT permitted to do something which is not clearly permitted.

Casper - Cluster Appliance for Scientific Parallel Execution and Rendering

Politecnico di Torino HPC project is an Academic Computing center which provides computational resources and technical support for research activities in accademic and didattical purposes.
The HPC project is officially managed by LABINF (Laboratorio Didattico di Informatica Avanzata) under supervision of DAUIN (Department of Control and Computer Engineering) which granted by Board of Directors.

Technical Specifications of Casper:

Architecture	Linux Infiniband-DDR MIMD Distributed Shared-Memory Cluster
Node Interconnect	Infiniband DDR 20 Gb/s
Storage Interconnect	Ethernet 10 Gb/s
Service Network Gigabit	Ethernet 2x 1 Gb/s (bonding 802.3ad)
CPU Family	AMD Bulldozer/Piledriver
CPU Model	Opteron 6276/6376 2.3 GHz (turbo 3.0 GHz) 16 cores
Sustained performance	4.360 TFLOPS
Peak performance	5.658 TFLOPS
Green500 Index	422.31 MFLOPS/W
Power Consumption	7 kW
Computing Cores	544
Number of Nodes	17
Total RAM Memory	2.2 TB DDR3 REGISTERED ECC
Working Storage	140 TB on RAID 6, throughput near 200 MB/s
OS	ROCKS Clusters
Scheduler	GridEngine

First Access

Your account has to be considered active from the moment you received confimation email containing your access credentials.

How get access to CASPER? It depends on which operating system is used by your computer. Assume that you are using Linux, Unix or OSX system, you can use a simple ssh client from any terminal with following:

user@pc$ ssh -X username@casperlogin.polito.it

to access CASPER please enter username and password associated to you by means of confirmation email.

If you are using a Windows system we suggest you to use PuTTY software (available at - http://www.putty.org) by setting the configurations as following:

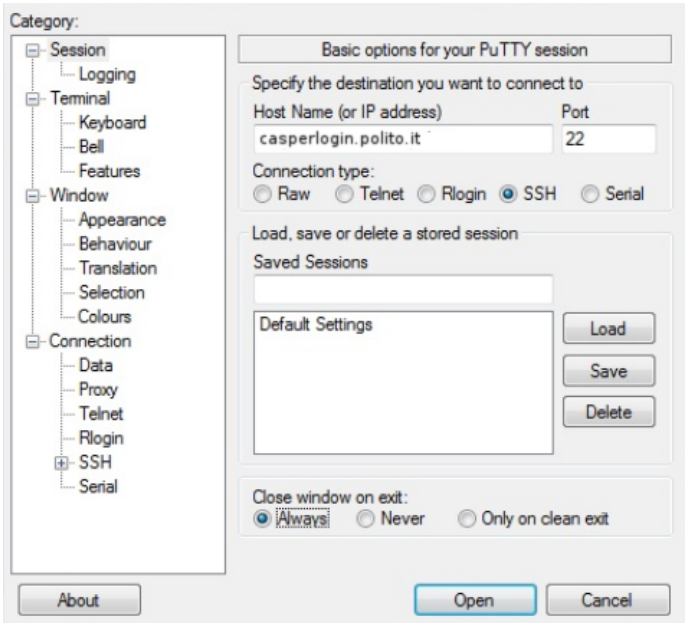

Figure 1: Putty - *PuTTY* Setting for CASPER access by Windows OS

We strongly suggest you to change your password after first log-in by usign command following:

username@login-0-0$ passwd

There is no limitation about length and composition of password, in any case is suggested to choose a password which is a combination of at least 8 characters, numbers, uppercase and lowercase letters.

Let us remind you that account sharing (the sharing of the same account among different users) is not permitted, moreover the person who made the request for the account has to take all the responsability to keep the credentials safe: choose a strong password, don't share your sensible data with others. Your account should be secure, in any case if you observe unexpected account behaviours, please contact HPC's Staff immediately.

File Transfer

After the access stage, the first showed directory is the user's *home*:

/home/username/

where the data have to be put in order to start a task and where the data will be written at the end of each task.
This directory can be accessed only by the owning user. It can contain at most 1 TB of data (eventually expandible).
To copy files/directories inside your main directory you can use *scp* command as following:

user@pc$ scp -r /path/to/local/dir/ username@casper.polito.it:/home/username

Instead, to copy files from CASPER on your local machine, use command following:

user@pc$ scp -r username@casper.polito.it:/home/username /path/to/local/dir/

It works similar as the *cp* command in Unix environment.

user@pc$ cp SOURCE DEST

Third option is to use the SFTP server which is available within the cluster.
For example, you can use the FileZilla software configured as following.

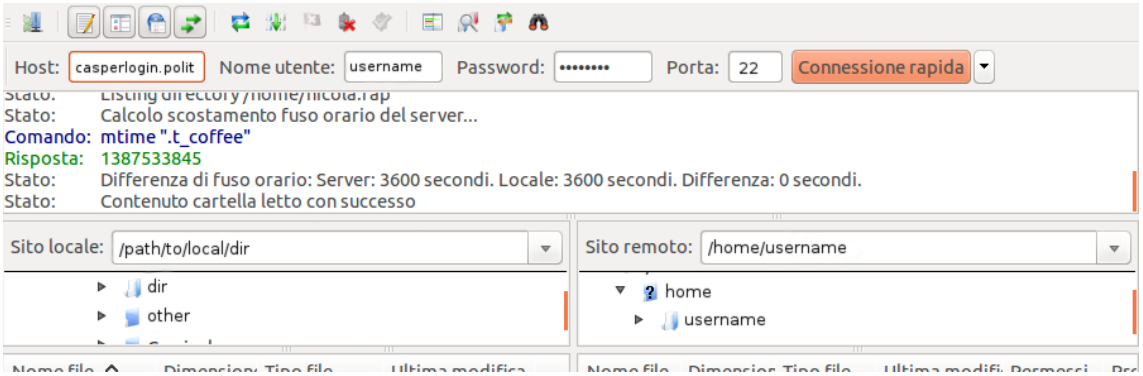

Figure 2: FileZilla configuration for file transfer on CASPER

Cluster Usage

The runnable processes on a cluster are "batch processes"; it means that are not interactive and execution of them can be postponed. Each task/job is composed by one or more processes that work together to achieve a certain result.
A job is executed after his schedulation; in order to schedule a job it must be inserted in a waiting list, managed by cluster, waiting to get the available resources held by oldest processes.
CASPER uses the Grid Engine scheduler to manage the waiting queues and availability of the computational shared resources.

There are two different queues for public access:

all.q Default queue for public access, which includes all nodes (544 Cores on 17 nodes); the maximum duration of a job is 10 days.

fast.q High priority queue, 8 Cores on 2 nodes; the maximum duration of a job is 60 min.

The jobs that requires low computation and low resources should be inserted in the *fast.q* which deals with jobs launched for testing activity or with very low execution time.

Controll and Submission of a Job
- qsub, qstat, qdel, qhost -

QSUB

All the jobs should be passed to the cluster scheduler through submission. It is possible to submit jobs using *qsub* command which recevices as argumnet a reference to a script file containing all the required information.
The script file, know as *qsub script*, is mainly composed by a set of directives and an execution command as should be appear on a command line. Some directives are mandatory(*), otherwise the job will stay in "*qw*" (waiting queue) indefinitely.

An example of this file is showed following:

[script.qsub]

```
#!/bin/bash
#$ -N Task_Name
#$ -M user_email@mail.com
#$ -m abes
#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -pe shared 32
#$ -l h_rt=HH:MM:SS
#$ -l virtual_free 16G
#$ -q all.q
example_task.bin
```

Directives:

-M	Email where the information on task will be sended.
-m	Events which cause an email notification (*es. abes* - abort, begin, end, suspend).
-cwd	If present cause the execution of task using the current dir as working dir.
-j y|n	Indicates if redirects or not stderr on output file.
-S	The preferred command interpreter.
-pe	(*) Specifies parallel environment in which the task will be executed. Any job requires a certain number of slots for execution, that can be "chosen" by the same node (max. 32) or more belonging to what is specified on this directive. The most commons are *shared*, *orte* and *mpirr*: **shared** - When it's required to execute many threads or processes on a single multi-core machine specifying the number of slot (*ex. -pe shared 16*). **orte** - When it's required to execute many parallel processes (usually MPI) on many nodes, but it tries to maximize the load on each one of them; Allocation Rule: "fill-up" (*ex. -pe orte 16*). **mpirr** - When it's required parallelize many processes using a certain number of slots, it tries to increase the distribution of each processes on many nodes; Allocation Rule: "round robin" (*es. -pe mpirr 16*). **mpisp** - When it's required to execute a certain number of MPI processes on each available node. Usefull for tasks that uses both MPI and OpenMP (example page 11).
-l h_rt	(*) Indicates the hard run time limit, which is about the time that processes needs to reach the end. This time is used by schedule to manage processes queue. This value must be less than 10 days (<240) for tasks on *all.q* queue.
-l virtual_free	Memory requirement for task execution.
-q	(*) Indicates the queue where each tasks has to be scheduled (for the most of the tasks use *all.q*).

QSTAT

In order to get information on the scheduled jobs with *qsub*, as *job-ID*, *status*, *queues* and the used slots number, it's possible to use command following:

$ qstat

```
job-ID  prior   name   user    state submit/start at    queue                  slots
-------------------------------------------------------------------------------
494180 0.51008 test   username  r  01/01/2015 00:00:00 all.q@compute-0-0.local  4
494182 0.51008 test   username  r  01/01/2015 00:00:00 all.q@compute-0-0.local  4
494183 0.51008 test   username  r  01/01/2015 00:00:00 all.q@compute-0-0.local  4
494184 0.51008 test   username  r  01/01/2015 00:00:00 all.q@compute-0-0.local  4
```

To obtain more information about state of each specific job you can use:

$ qstat -j {job-ID}

```
$ qstat -j 123456
==============================================================
job_number:                 123456
exec_file:                  job_scripts/123456
submission_time:            Thu Jan 01 00:00:00 2015
owner:                      username
uid:                        000
group:                      usergroup
gid:                        000
sge_o_home:                 /home/username
sge_o_log_name:             username
sge_o_path:                 ...
sge_o_shell:                /bin/bash
sge_o_workdir:              /home/username/...
sge_o_host:                 casper
account:                    sge
cwd:                        /home/username/...
merge:                      y
hard resource_list:         h_rt=720000
mail_list:                  username@casper.local
notify:                     FALSE
job_name:                   jobname
jobshare:                   0
hard_queue_list:            all.q
shell_list:                 NONE:/bin/bash
script_file:                run
parallel environment:       orte range: 32
usage    1:                 cpu=185:03:51:38, mem=2380221.15046 GBs, ...
scheduling info:            ...
```

To obtain information about the entire cluster and the jobs executing on it, use:

*$ qstat -u **

QDEL

To stop and remove a job from a queue, use:

$ qdel {job-ID}

QHOST

The *qhost* command allows you to know state of nodes on the cluster at a given time:

$ qhost

```
HOSTNAME                ARCH         NCPU  LOAD  MEMTOT  MEMUSE  SWAPTO  SWAPUS
----------------------------------------------------------------------------------
global                  -               -     -       -       -       -       -
compute-0-0             linux-x64      32  2.00  126.2G    2.1G 1000.0M  394.7M
compute-0-1             linux-x64      32  2.41  126.2G   32.0G 1000.0M  601.8M
compute-0-10            linux-x64      32 20.02  126.2G    5.1G 1000.0M  166.9M
compute-0-11            linux-x64      32  1.03  126.2G    3.8G 1000.0M  210.4M
compute-0-12            linux-x64      32  2.02  126.2G    5.7G 1000.0M  336.6M
compute-0-13            linux-x64      32  2.04  126.2G    5.2G 1000.0M  174.6M
compute-0-14            linux-x64      32 38.39  126.2G   25.0G 1000.0M  398.0M
compute-0-15            linux-x64      32 32.08  126.2G   22.5G 1000.0M  241.5M
compute-0-16            linux-x64      32 38.57  126.2G   25.9G 1000.0M  399.6M
compute-0-2             linux-x64      32  5.19  126.2G    2.4G 1000.0M  443.1M
compute-0-3             linux-x64      32  3.20  126.2G    4.6G 1000.0M  437.3M
compute-0-4             linux-x64      32  5.21  126.2G    5.0G 1000.0M  349.7M
compute-0-5             linux-x64      32 14.96  126.2G    4.0G 1000.0M  154.8M
compute-0-6             linux-x64      32 21.08  126.2G    9.1G  996.2M   10.7M
compute-0-7             linux-x64      32  6.05  126.2G    8.0G  996.2M  512.6M
compute-0-8             linux-x64      32  6.01  126.2G   12.1G  996.2M  108.0M
compute-0-9             linux-x64      32  1.00  126.2G    4.0G 1000.0M  315.7M
```

Moreover is possible to get information about distribution of a user's jobs on the cluster nodes by using:

$ qhost -u username

MODULES

The software *modules* (*Environment Modules Project*) enables dynamic modification of the environment variables during a user session. The use of this software is strongly recommended in the *qsub script* because it provides an easy way to use different versions of the same application; it allows user to export environment variables.

You can get a list of all the available modules through the command following:

username@login-0-0$ module avail

The modules can be loaded (*load*) or unloaded (*unload*) from current user session. For example, to load the modules required to enable *openmpi over infiniband*, you can use:

username@login-0-0$ module load openmpi_ib

To remove all the current loaded modules from the session, you can use the command:

*username@login-0-0$ module unload **

To get the list of currently loaded modules you can use:

username@login-0-0$ module list

To be effective, in the most of the cases, the modules' commands should be directly added in the qsub script.

For example, to use *openmpi over infiniband* it's required to add on the qsub script following line:

module load openmpi_ib

GANGLIA

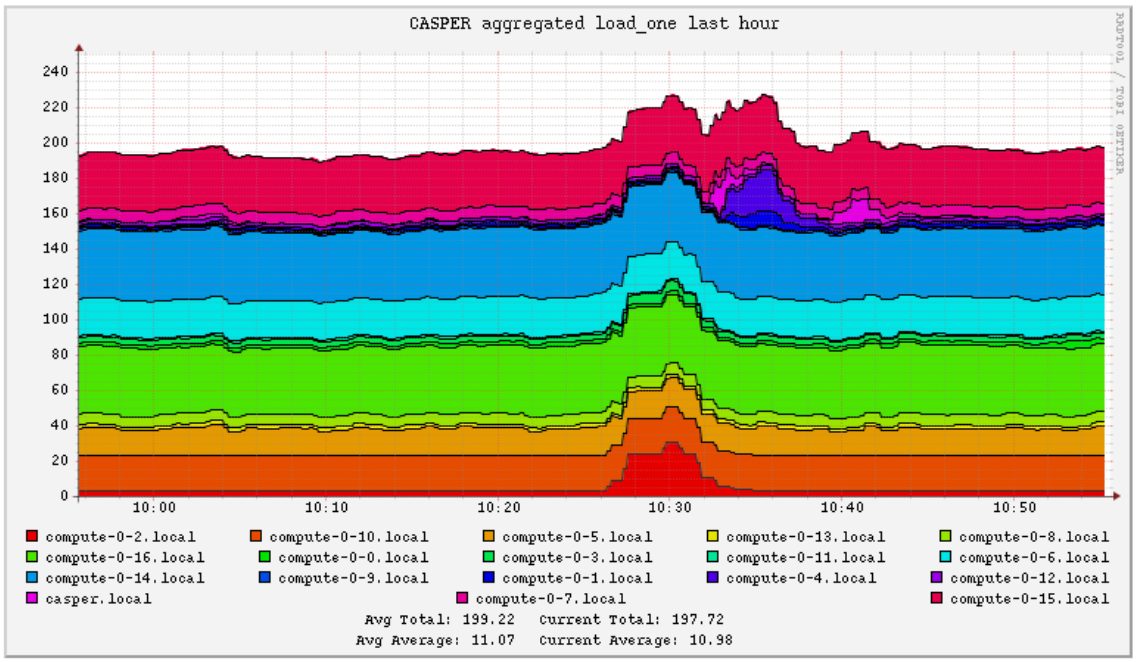

Figure 3: Ganglia - Ganglia example

Ganglia is a web interface that provides information about the cluster's activities. It available at following link:

http://casper.polito.it/ganglia/

APPENDIX A

It follows a list of script templates for some of the most used applications installed on CASPER.

Matlab [matlab.qsub]

```
#/bin/bash
#$ -N test_matlab
#$ -M user_mail@mail.com
#$ -m abes
#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -pe shared 16
#$ -q all.q
#$ -l h_rt=00:10:00

module load matlab

matlab -r test_matlab
```

Blender [blender.qsub]

```
#!/bin/bash
#$ -N test_blender
#$ -j y
#$ -m beas
#$ -M user_mail@mail.com
#$ -S /bin/bash
#$ -pe shared 32
#$ -q all.q
#$ -l h_rt=1:00:00
#$ -cwd

module load blender-2.69

blender -noaudio -b BMW1M-MikePan.blend.1 -o Cycles_Benchmark -F PNG -f 1 -t 32
```

Comsol [comsol.qsub]

```
#!/bin/bash
#$ -N comsol_test
#$ -m abe
#$ -M user_mail@mail.com
#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -pe shared 16
#$ -q all.q
#$ -l virtual_free=64G
#$ -l h_rt=120:00:00

module load comsol50

comsol -np $NSLOTS batch -inputfile input_file.mph -outputfile output_file.mph
```

Cplex [cplex.qsub]

```
#/bin/bash
#$ -N test_cplex
#$ -M user_mail@mail.com
#$ -m abes
#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -pe shared 8
#$ -q all.q
#$ -l h_rt=01:00:00
#$ -l virtual_free=32G

module load cplex

cplex < problem.lp.cplex > problem.cplex.out
```

problem.lp.cplex

```
read problem.lp
set timelimit 41472
set mip display 0
set mip tol mipgap 0
set parallel 1
set threads 8
set workmem 8192
set mip limits treememory 32696
optimize
quit
```

MPI PROCESSES

MPI parallel processes can be distributed on Casper with two different allocation rules: *fill-up* or *round-robin*.

Fill-up tries to "fill" the node, exploiting the available (per node) resources and accelerates processes executions. It is enabled on *orte* Parallel Environment (PE).

Round-robin tries to spread processes over all available nodes, that is, each process potentially can reach more resources. It is enabled on *mpirr* Parallel Environment (PE).

MPI (FILL-UP) [mpi.qsub]

```
#$ -M user_mail@mail.com
#$ -m abes
#$ -N mpi_quicksort
#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -pe orte 96
#$ -q all.q
#$ -l h_rt=5:00:00
#$ -l virtual_free=16G

module load openmpi

time /opt/openmpi/bin/mpirun -np 96 mpi_quicksort.x input_21GB.txt output.txt
```

MPI (ROUND-ROBIN) [mpirr.qsub]

```
#!/bin/bash
#$ -M user_mail@mail.com
#$ -m abes
#$ -N mpi_quicksort
#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -pe mpirr 16
#$ -q all.q
#$ -l h_rt=5:00:00
#$ -l virtual_free=16G

module load openmpi

time /opt/openmpi/bin/mpirun -np 16 mpi_quicksort.x input_21GB.txt output.txt
```

In some cases it is required to exploit a certain number of MPI process per node (e.g. one process on each node). It is possible to use the *mpisp* Parallel Environment (PE) for this scope.

The mechanism is exploited by mean of a script.

The following data have to be passed at *mpirun* command:

- a list of available host (e.g. *-machinefile $JOB_ID.machines*)
  - *$JOB_ID.machines* is a file automatically generated by the Scheduler for this purpose;
- the number of processes involved (e.g. *-np $(($NHOSTS*$NPN))*
  - *$NHOST*, is the number of available host on a given moment;
  - *$NPN*, is the number of processes per node;
- number of processes per node (e.g. *-npernode $NPN*)

Finally the script shall appear as following:

..... MPI (NUMBER OF PROCESSES PER NODE) [mpisp.qsub] .....

```
#!/bin/bash
#$ -S /bin/bash
#$ -pe mpisp 16
#$ -cwd
#$ -V
#$ -N mpitest
#$ -q all.q
#$ -l h_rt=1:00:00

#############################################################
NPN=1 # set here the number of processes per node
#############################################################

module load openmpi_ib

# $JOB_ID.machines, file containing the avaible host list, generated by sge
# $NHOSTS, number of avaible hosts passed by sge

mpirun -machinefile $JOB_ID.machines -np $(($NHOSTS*$NPN)) -npernode $NPN mergesort
```

.....

APPENDIX B

For advance tasks which requires multiple execution of the same program with different parameters, we suggest you to use following script which is able to generate multiple qsub script file by starting from a text file containing all the parameters.

[generate.sh]

```
#!/bin/bash
#
list=$(cat ./parameters.csv)
#
for i in $list
do
parameter1=$(echo $i|cut -f 1 -d ‘,’)
parameter2=$(echo $i|cut -f 2 -d ‘,’)
cat << EOF > ./test-$parameter1-$parameter2.qsub
#/bin/bash
#$ -N task-$parameter1-$parameter2
#$ -M user_mail@mail.com
#$ -m abes
#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -pe shared 32
#$ -q all.q
#$ -l h_rt=01:00:00
#$ -l virtual_free=32G
example.bin  $parameter1 $parameter2
EOF
done
```

[parameters.csv]

```
A,1
A,2
B,1
B,2
```